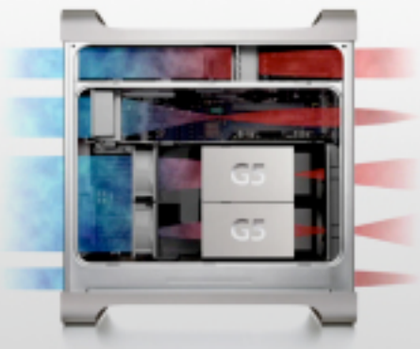
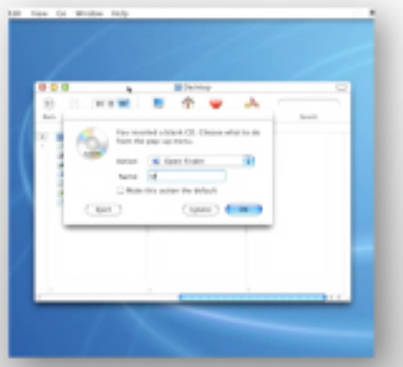
USB Update

Session 503





Introduction

Craig Keithley
I/O Technology Evangelist
keithley@apple.com

Introduction

- Share new information on what the team has been doing for the last year
- Full support for USB 2.0 Specification
- New APIs since we last met
- Administrivia

What You Will Learn

- Introduction to USB 2.0
- How USB 2.0 is implemented
- USB 2.0 Isochronous transactions
- USB 2.0 Hubs
- API Updates

A Gentle Introduction to USB 2.0

Barry Twycross
USB Software Team
barryt@apple.com

USB 2.0

- What it is, and what it is not
- What it means to you
- How it is done
- High Speed Isoch APIs

What Is USB 2.0

- It is USB
  - Compatible with USB 1.1
    - Includes almost all of the USB 1.1 spec
- Adds High Speed (480 Mb/s)

What USB 2.0 Is Not

- USB 2.0 != High speed
  - Full and Low (“Classic”) speeds are USB 2.0
  - USB-IF frowns on confusing “High speed” and “USB 2”
    - Marketing tricks (boo, hiss)
- USB 2.0 is NOT inefficient!
  - Speeds are segregated
    - Confusion about this
  - Performance tweaks
    - 512 byte bulk packets, NYET handshake, PING protocol

What it Means to Device Developers

- Very little (Classic speeds)
  - Change “bcdUSB” version number in Device Descriptor
- However:
  - STALL unknown requests (As you should already)
    - Especially the Device Qualifier Descriptor request
  - Isoch endpoints must use no bandwidth by default
  - Change in low speed cabling requirements
    - Add inner shield and drain, recommends shield and twists

What it Means to Device Developers

- A lot (High Speed)
  - Can make things go much faster, with redesign
    - 24 MByte/s now for Isoch and Interrupt
      - Uncompressed video is now possible
    - 20–24 MByte/s now for bulk
      - Depends on host controller implementation
  - Theoretical 35–40 MByte/s for Bulk

What it Means to Driver Developers

- Very little
  - High Speed device just works
    - Should not care about the speed
    - Just transfer the data
- Except for Isoch
  - Isoch timing is all different
    - 125 µs micro-frames
  - Up to 3072 bytes per micro frame (also for Interrupt)

How Is it Done?—On the Bus

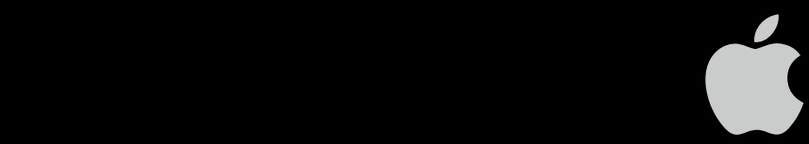
- EHCI (Enhanced Host Controller Interface)
  - New controller standard
    - Support in Panther
  - Companion controllers (Good old OHCI)
    - Automatic routing to correct controller
    - Works as Classic speed controller without EHCI driver
- Magic hubs
  - Most of a host controller in there!

How Is it Done?—Getting Started

- Hardware
  - New Macintosh with built-in high-speed controller
  - PCI or PC Card (with OHCI companion controllers)
- Software
  - Panther
  - Make it work full speed first
- Tools
  - High Speed Analyzer
    - CATC Advisor, Catalyst SBAE-20, Data Transit

High Speed Isoch

- High Speed Isoch is different
  - To the device and to the driver
  - 125µs Micro-Frames, 8 per frame
  - Up to 3072 byte transfers per micro-frame, new PIDs
  - Frequency need not be 1 transfer per (micro) frame
    - No devices, we have not implemented this yet
- High Speed Isoch is the same
  - Same APIs used
  - Some things re-interpreted



High Speed Isoch Packets

Packet sizes and sequence

Isoch IN uses Data2/1/0 PIDs (In that order)

1 Transaction, < 1024 bytes:	DATA0		
2 Transaction, 513–1024 bytes each	DATA1	DATA0	
3 Transaction, 683–1024 bytes each	DATA2	DATA1	DATA0

Isoch OUT uses MData and DataX PIDs

1 Transaction, < 1024 bytes:	DATA0		
2 Transaction, 513–1024 bytes each	MDATA	DATA1	
3 Transaction, 683–1024 bytes each	MDATA	MDATA	DATA2

Interrupt uses same old Data0, Data1 PIDs

High Speed Isoch APIs

- Frame is now Transfer opportunity

```
IOReturn (*ReadIsochPipeAsync)(
    void *  self,
    UInt8   pipeRef,
    void *  buf,
    UInt64  frameStart,    <-- Note still specified as frame
    UInt32  numFrames,     <-- Now number of transfers
    IOUSBIsocFrame * frameList,   <-- Now transfer list.
    IOAsyncCallback1 callback,
    void *          refcon );
```

Also GetFrameListTime

Summary

- USB 2.0 runs faster (High Speed)
- USB 2.0 is just like 1.1 if you do not use High Speed
- Do not worry about the speed if you do not have to
  - You have to worry if you are doing Isoch
- High Speed Isoch is different
  - APIs are the same, interpretation is different

High Speed Hubs with USB 2.0

Rhoads Hollowell
USB Software Team
rhoads@apple.com

USB 2.0 High Speed Hub

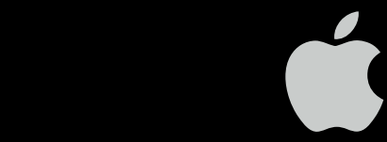
- Where all the magic happens
- The root hub is special because of the companion controller
  - Multiple companion controllers
- Transparent to most driver writers
  - Hub drivers do have to know a little bit about it

USB 2.0 High Speed Hub (Cont.)

- Can cause problems if not configured properly
  - Becomes a USB 1.1 hub when plugged into another USB 1.1 hub
  - All downstream devices are then in Full Speed/Low Speed mode
- The magic is in the Transaction Translator (TT)

Transaction Translator (TT)

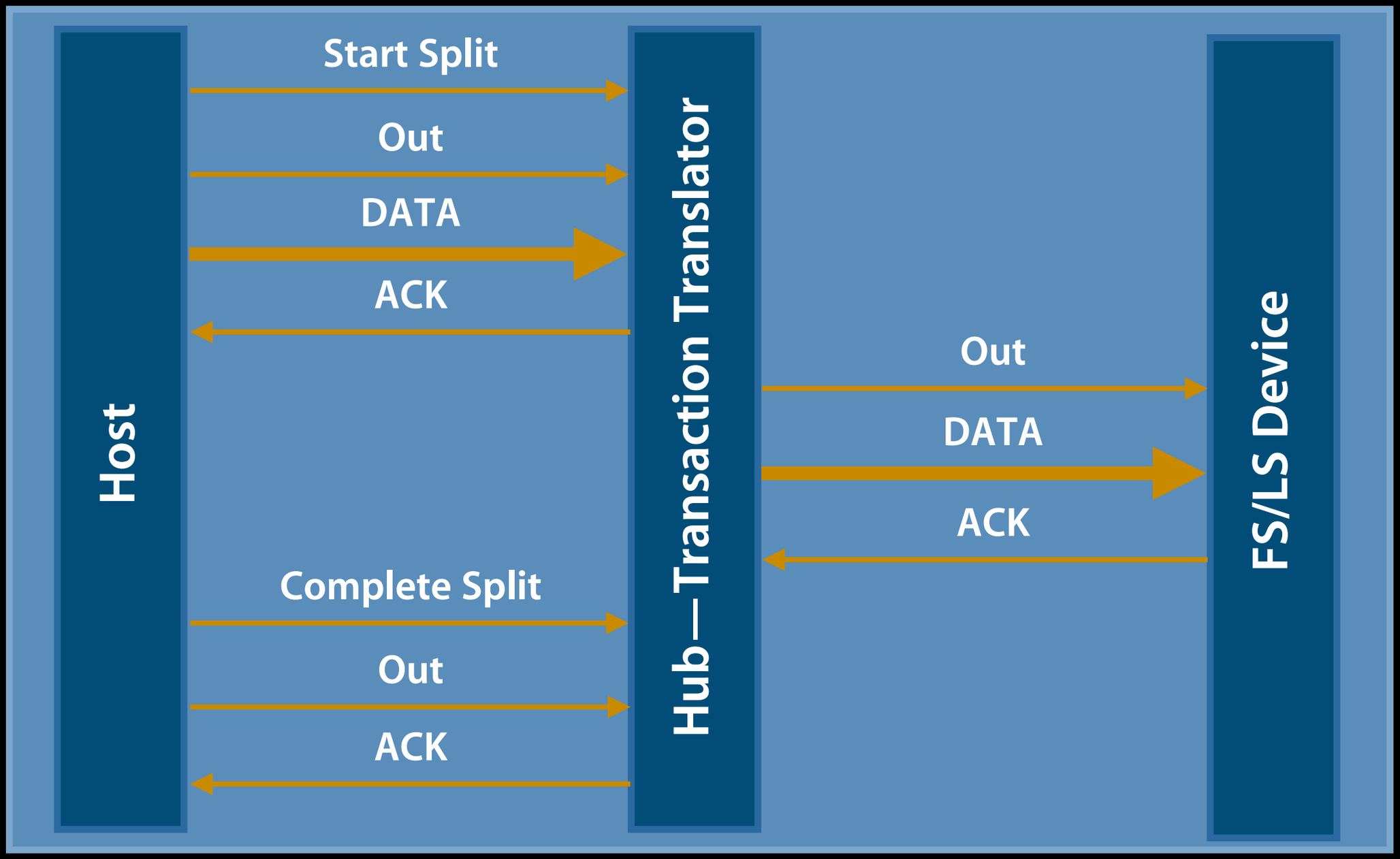
- At least one per High Speed hub
- Can have one per port in the hub
- One Full Speed (FS) controller's worth of bandwidth for each TT (within reason)
  - Multi-TT hubs can have more FS Isoch devices than single-TT hubs
- Root Hub does not use TTs—passes the request off to the companion controller instead
- TT is a store-and-forward unit for split transactions



Split Transactions

- High Speed transfer to hub (TT)
- Full/Low Speed transfer to FS/LS device
- Transaction between Host Controller and TT is in two parts
  - Start Split
  - Complete Split
  - Other HS transactions can happen in-between
- Can have significant performance issues

Split Transaction Example



The time from the FS/LS ACK to the Complete Split transaction can cause performance degradation compared to FS/LS on a USB 1.1 controller

Single TT vs. Multi-TT hubs

- Each TT essentially acts as a Full Speed bus
- Especially impacts Isochronous endpoints
  - Example: a 980 byte/ms camera
    - Single TT hub can have one of them
    - Multi TT hub can have one per TT
  - Another example: camera with mic and speakers
    - Might be able to have both of them in a multi TT hub
    - or have lower quality in a single TT hub or in a USB 1.1 hub
- More TTs allow better performance with bulk as well

Root Hubs are Different

- We have companion controllers
- No need for TT!
- EHCI driver (if it exists) determines which controller (EHCI or companion OHCI) gets the device
- So FS/LS devices plugged into root hubs do not use split transactions

Summary

- Split transactions allow for seamless transition to USB 2.0
  - But can cause performance issues
  - Performance critical FS devices should probably stay on the OHCI bus
- A FS hub plugged into a root hub will provide a FS bus
- A HS hub plugged into a root hub will provide a HS bus

API Update

Fernando Urbina
Senior Software Engineer
nano@apple.com

API Update

- Kernel and user updates
- Get the IOUSBLib and IOUSBFamily versions
  - `GetIOUSBLibVersion( NumVersion *iousbLib, NumVersion *iousbFamily)`
- High Speed USB Related
  - `GetFrameListTime( UInt32 *microsecondsInFrame )`
    - Returns either `kUSBFullSpeedMicrosecondsInFrame` or `kUSBHighSpeedMicrosecondsInFrame`
  - `GetBusMicroFrameNumber( UInt64 *microFrame, AbsoluteTime *atTime)`

API Update

- Simpler API to recover from stalls
  - `ClearPipeStallBothEnds( UInt8 pipeRef )`
    - Clears the stall on both the device and the controller
    - Necessary to synchronize the data toggle
- Isochronous related
  - 3 new APIs that allow you (and us) to better manage USB bandwidth
  - IOUSBPipes will now be created even if there is not enough bandwidth available (change from earlier)

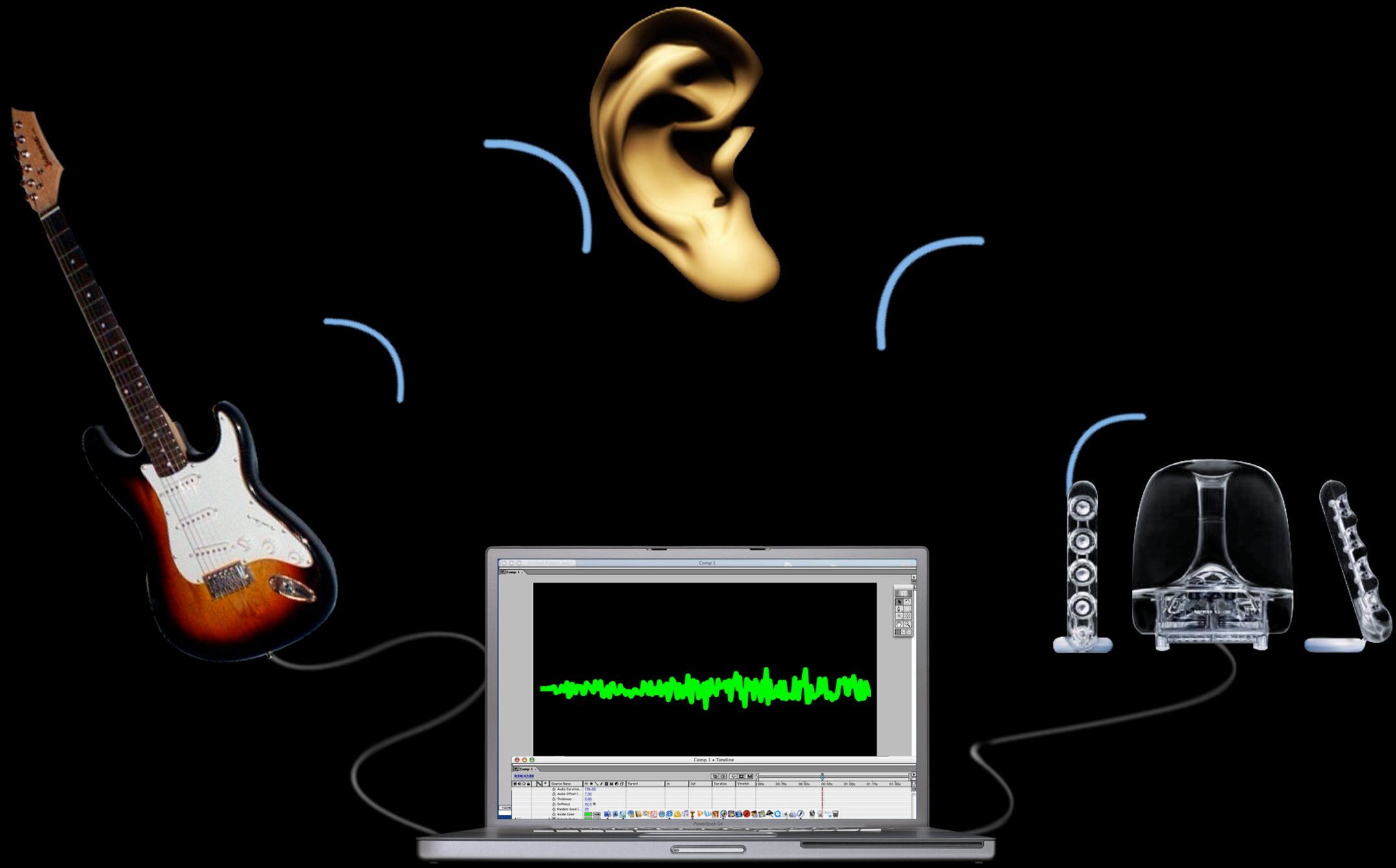
API Update

- Creating an isochronous endpoint and managing bandwidth
  - `GetBandwidthAvailable()`
  - `GetEndpointProperties()`
  - `SetAlternateInterface()`
  - `GetPipeProperties()`
    - If `maxPacketSize` = 0, then there was not enough bandwidth for pipe
  - `SetPipePolicy()`

The Latency Problem

- What is latency
  - Delay between the hardware interrupt and callback
  - Callbacks happen on USB Workloop
    - Higher priority threads (audio, UI)
    - Can be upwards of 80 ms!
- Use multiple transfers to keep pump primed
  - Sometimes, not enough

Low Latency Scenario



Low Latency Isoch Solution

- The data is in the user's buffer as soon as the USB controller completes the transaction
- Only need to update the number of bytes transferred
- Client can peek into the buffer at the expected completion time to get the data
- FrameList will have the updated `frActCount` and time stamp
  - Do processing at primary interrupt time
  - Update structure with a time stamp

Low Latency Isoch API

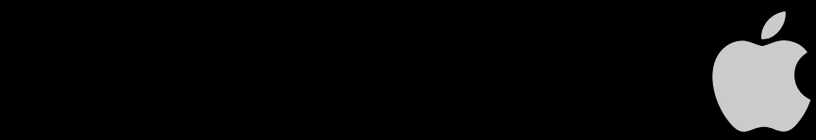
- Four new calls
  - LowLatencyCreateBuffer(…)
  - LowLatencyDestroyBuffer(…)
  - LowLatencyReadIsochPipeAsync(…)
  - LowLatencyWriteIsochPipeAsync(…)
- IOUSBLib will create and destroy the buffers
  - Client calls in to create the data and frame list buffers
  - Can use one big buffer or multiple ones
  - Afterwards, client needs to call in to destroy the buffer

Low Latency Isoch API

- API for read and writes:
-

```
LowLatencyReadIsochPipeAsync(
            UInt8                           pipeRef,
            void *                          buf,
            UInt64                          frameStart,
            UInt32                          numFrames,
            UInt32                          updateFrequency,
            IOUSBLowLatencyIsocFrame *      frameList,
            IOAsyncCallback1                callback,
            void *                          refcon)
```

- updateFrequency: how often we update the frame list at primary interrupt time



Low Latency Isoch API

- New Low Latency Frame List
-

```
struct IOUSBLowLatencyIsocFrame {
    IOReturn            frStatus;
    UInt16              frReqCount;
    UInt16              frActCount;
    AbsoluteTime        frTimeStamp;
};
```

- HeaderDoc for IOUSBLib has detailed usage
- Do not abuse it

USB Implementers Forum Update

- Video Device Class Specification nearing release
  - Covers large range of video devices, from webcams to high end camcorders
  - Apple plans to provide a class driver
  - Full and High speed devices
  - Bulk and isoch pipes
  - Still image
  - Video in and video out

USB Implementers Forum Update

- Interface Association Descriptor
  - Change Notice to USB 2.0 spec
  - Allows a device to describe which interfaces are associated with the same device function
  - Video Class Spec will use it to associate Video Control and Video Streaming interfaces
  - Investigating support for this descriptor

Debugging Tools update

- Log versions of IOUSBFamily available on the web
- Do NOT try to boot with an unstripped version of IOUSBFamily
- In order to generate symbols, use:
  - `% sudo pbxbuild install UNSTRIPPED_PRODUCT=YES`
- Setup your boot-args to allow for 2 machine debugging:
  - `% nvram boot-args=”debug=0x144”`
- Use the CHUD tools

Administrivia

- Repository is not live anymore
  - Still open-sourced
  - Source is released after OS update
  - EHCI driver will be in Panther release

Summary

- GetVersion API
- High Speed API
- Bandwidth management
- Low Latency
- Video Class Spec
- Repository not live anymore

For More Information

Reference Library

Documentation

Accessing Hardware From Applications
http://developer.apple.com/techpubs/macosx/Darwin/General/AccessingHardware

I/O Kit Fundamentals
http://developer.apple.com/techpubs/macosx/Darwin/General/IOKitFundamentals/index.html

Writing an I/O Kit Device Driver
http://developer.apple.com/techpubs/macosx/Darwin/General/WritingDeviceDriver/index.html

Working with USB Device Interfaces
http://developer.apple.com/techpubs/macosx/Darwin/IOKit/DeviceInterfaces/USB/USBBook/index.html

Reference Library

Documentation (Cont.)

User-mode USB Device Arbitration

http://developer.apple.com/techpubs/macosx/Darwin/IOKit/DeviceInterfaces/USB/ClassicUSBDeviceArb.html

Reference Library

Technical Notes

TN2024: Understanding and Debugging Kernel Panics
http://developer.apple.com/technotes/tn2002/tn2063.html

Technical Q&As

QA1076: Tips on USB Driver Matching for Mac OS X
http://developer.apple.com/qa/qa2001/qa1076.html

QA1087: Issues with boot time KEXT loading
http://developer.apple.com/qa/qa2001/qa1087.html

QA1075: Making sense of I/O Kit error codes

http://developer.apple.com/qa/qa2001/qa1075.html

Wrap Up

Roadmap

100 Kernel Extension Programming... Designing KEXTs for optimal performance	**Russian Hill** **Mon., 3:30pm**
107 Kernel Programming Interfaces Preview of upcoming APIs for KEXTs	**Marina** **Wed., 10:30am**
506 CHUD Performance Optimization... Collect Data and Measure Performance	**Marina** **Wed., 2:00 pm**
FF006 FireWire and USB Feedback Forum Tell us what you think	**North Beach** **Wed.,3:30pm**
509 HID and Force Feedback Supporting HID and using Force Feedback	**Marina** **Fri., 9:00am**

Who to Contact

Craig Keithley
I/O Technology Evangelist
keithley@apple.com

Barry Twycross
USB Software Team
Barryt@apple.com

Rhoads Hollowell
USB Software Team
rhoads@apple.com

Fernando Urbina
USB Software Team
nano@apple.com

http://developer.apple.com/wwdc2003/urls.html

Q&A

USB Software Team

usb@lists.apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference